Log Management in NFV Service Orchestration

Engin Zeydan, Jorge Baranda, Josep Mangues-Bafalluy, Ricardo Martínez, Luca Vettori

Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Castelldefels, Spain, 08860

Emails: {engin.zeydan, jbaranda, josep.mangues, rmartinez, lvettori}@cttc.cat

***Abstract*—Measuring several relevant metrics related to Network Function Virtualization (NFV) service lifecycle management in real time brings an enhanced monitoring of the operation of the network service orchestrator (SO) and the NFV infrastructure. In this demonstration, we integrate a complete data engineering pipeline in an operational management and orchestration stack (that of EU 5Growth project) to analyze lifecycle management metrics in real-time, in this case the network service instantiation time related metrics. In our demonstration, a data connection module instance continuously monitors the NFV SO log files and sends the log changes to the data ingestion layer, where log files are temporarily stored to be fetched by Apache Spark jobs. After utilizing Spark jobs to cleanse the log files and to obtain the service instantiation times, the metrics are sent back to the data ingestion layer to be transferred to the Elasticsearch (ELK) stack for data indexing and visualization purposes. Furthermore, the statistical information of network service instantiation (in total and its components) of studied metrics inside the network can also be profiled via a separate data analysis layer connected to the ELK stack.**

***Index Terms*—log parsing, service orchestrator, data pipeline, monitoring.**

I. Introduction

End-to-end network lifecycle management (LCM) is of vital importance for service providers. Robust network LCM requires monitoring several network service related metrics simultaneously. For this reason, a scalable, reliable and fault-tolerant data engineering infrastructure needs to be built to monitor several service instantiation-related metrics during the end-to-end service orchestration process.

During the network service instantiation process, there are many internal steps occurring between the building blocks of the database module, service orchestration engine, resource orchestration engine and core management and orchestration (MANO). These steps are just some examples of the components that can be profiled. The paper in [1] details the Service Orchestrator (SO) functionality, implementation and operation. The EU 5Growth (5Gr) project [2] features three main architectural building blocks in its MANO stack, namely the Vertical Slicer (5Gr-VS), the SO (5Gr-SO) and the Resource Layer (5Gr-RL) for service and resource level management. The 5Gr-SO block oversees the end-to-end orchestration and the lifecycle management of NFV network services.

In this demo, we provide a tool for log parsing in ETSI NFVOs that allows monitoring real-time or in batch lifecycle management process components. More specifically, the demo shows this capability for service instantiation using the 5Gr-SO logs of various service instantiation-related metrics such as total instantiation time, the time to recollect the information from the RL, time spent in the Core MANO wrapper module during the instantiation process, etc [1]. For the demo, we monitor 21 of such network service instantiation metrics.

II. System Architecture

The proposed architecture is given in Fig. 1 which is a data pipeline that manages the monitoring of instantiation time of services initiated by SO in both batch and real-time mode by monitoring log changes inside the SO. We use open source tools and platforms such as Apache Flume [3] for data connection, Apache Kafka [4] (for data ingestion), Apache Spark [5] (for cleansing real-time data) and Elasticsearch (ELK) stack [6] (for data visualization) to build the data engineering pipeline. The architecture is composed of five main modules: **(a) Data Connection**, **(b) Data Ingestion** **(c) Data Pre-processing** **(d) Data Visualization** and **(e) Data Analysis**.

In the **Data Connection** module, the Apache Flume application periodically monitors the changes in logs of 5Gr-SO and sends data to the data ingestion layer. In the **Data Ingestion** module, the Kafka application ingests log messages transmitted from Apache Flume and the log messages fetched by Kafka are temporarily stored in the Kafka broker under a given topic name. In **Data Pre-processing** module, the main task is to parse and cleanse the incoming log messages from the Kafka topic and to extract the relevant metric values by using Spark libraries [5] in real-time. Finally, in the **Data Visualization** module, the values of the calculated metrics are present in the dashboard screen of Kibana. ElasticSearch is using Logstash subscribed to the Kafka topic to gather the metrics. For **Data Analysis**, we use Python libraries, sklearn, pandas and seaborn to statistically analyze the data that is indexed by Elasticsearch via a connection to the search engine using the Python Elasticsearch Client[1].

This work was partially funded by EU Commission H2020 5Growth project (Grant No. 856709), Spanish MINECO grant TEC2017-88373-R (5G-REFINE), Generalitat de Catalunya grant 2017 SGR 1195 and Frontier R&D Laboratories Support Program with project no: 5169902.

[1]https://elasticsearch-py.readthedocs.io/en/v7.12.1/, Accessed: April-2021

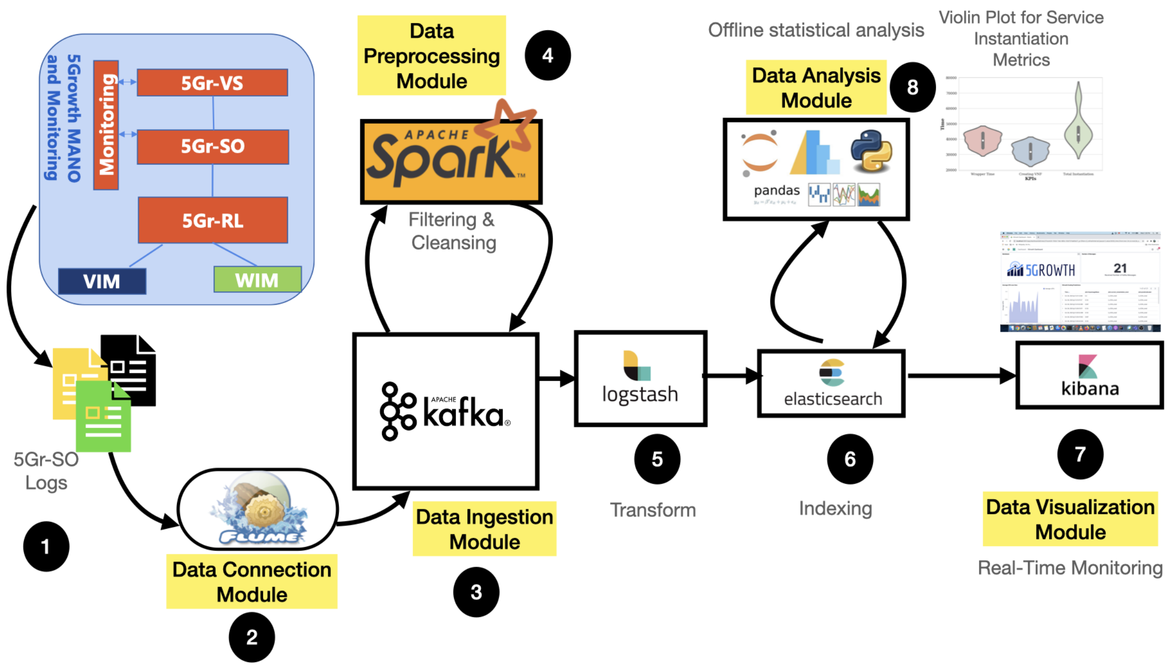

Fig. 1: General architecture for integration of log-parser in an end-to-end mobile network management architecture of 5Growth including demonstration setup and workflow structure.

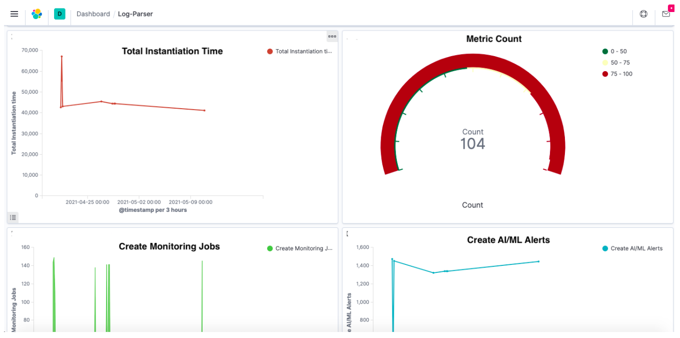

Fig. 2: Kibana dashboard for 5Gr-SO log monitoring.

III. Demonstration Workflow

We demonstrate the working principle of the system in steps (1)–(8), as given in Fig. 1. The general workflow is as follows: First, the 5Gr-SO starts logging the whole service instantiation process in step-(1). In step-(2), the Apache Flume application performs periodic log message readings of the 5Gr-SO and submits the log changes into the Kafka broker illustrated in step-(3). Later, every predefined duration of time, e.g., 1 second, a Spark job in the data analysis and processing manager of the data pipeline, collects data from the Kafka broker and cleanses the log data in step-(4). After obtaining the enriched and cleansed results of the Spark job, they are sent back to the Kafka broker so that the results can be picked up by the data visualization manager in step-(4). In step-(5), (6) and (7) we have the ELK stack, where data in Kafka is collected by the Logstash module in step-(5) and is pushed to Elasticsearch in step-(6) so that the output of log time differences of the system can be visualized by Kibana in step-(7). Finally, the log data present in ELK stack is statistically analyzed in step-(8). The dashboard in Fig. 2 displays the number of processed Kafka topics in the data engineering pipeline by Apache Spark and the results of the service instantiation time values of each metrics.

References

[1] J. Mangues *et al.*, "5g-transformer service orchestrator: design, implementation, and evaluation," in *2019 European Conference on Networks and Communications (EuCNC)*, pp. 31–36, IEEE, 2019.

[2] X. Li *et al.*, "5growth: An end-to-end service platform for automated deployment and management of vertical services over 5g networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021.

[3] Apache Flume, "A distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data." https://flume.apache.org/, 2021. [Online; accessed May-2021].

[4] Apache Kafka, "An open-source distributed event streaming platform." https://kafka.apache.org/, 2020. [Online; accessed Nov.-2020].

[5] Apache Spark, "Unified analytics engine for big data." https://spark.apache.org/, 2020. [Online; accessed Nov.-2020].

[6] Elasticsearch, "Open source search." https://www.elastic.co/, 2020. [Online; accessed Nov.-2020].